1.
Građa računala

1. Funkcijski opis računala, von Neumannov model računala

U digitalnoj elektronici naučili smo na koji se sve način mogu izgraditi pojedinačni digitalni elektronički sklopovi koji služe pohranjivanju i obradbi raznovrsnih podataka. Naučili smo da se pojedinačni bitovi mogu pohranjivati u bistabile, a nizovi bitova u registre. Nizovi bitova mogu se promatrati kao kôdovi za predstavljanje brojeva i znakova. Znamo, nadalje, da se s pomoću logičkih sklopova mogu izgraditi sklopovi pretvorbe kodova, za zbrajanje i oduzimanje brojeva i općenito za obavljanje raznovrsnih aritmetičkih operacija s binarno kodiranim brojevima. Prisjetimo se tog gradiva koje smo izučavali u prethodnim razredima i neka nam pri ruci bude udžbenik koji to gradivo obrađuje[1]. Tijekom izučavanja gradiva koje slijedi morat ćemo u nju češće zaviriti.

Od digitalnih elektroničkih sklopova mogu se sastaviti manji ili veći elektronički uređaji koji obavljaju neku konkretnu nama korisnu funkciju. Prije izgradnje takve elektroničke naprave ili uređaja moramo unaprijed utvrditi — do zadnjeg detalja — svaku pojedinačnu funkciju. Nakon što smo sve to detaljno utvrdili, možemo pristupiti najprije projektiranju, a zatim izgradnji naprave.

Pokazalo se da se za izgradnju raznovrsnih naprava može koristiti već gotovo sklopovlje sastavljeno u obliku računala, s tim da se pojedine funkcije u njemu oživotvoruju njegovim programiranjem. Današnje mogućnosti mikroelektroničke tehnologije omogućuju ekonomičnu proizvodnju računalnog sklopovlja, tako da se mogu izgraditi raznovrsna računala prilagođena po svojim mogućnostima, a i po cijeni, vrlo raznolikim potrebama.

Bez obzira da li se radi o malom računalu ugrađenom u daljinski upravljač televizora, o osobnom računalu ili o velikom računalu koje služi za neka složena

[1] Stanko Paunović, *Digitalna elektronika*, Školska knjiga, Zagreb, 1995

aritmetičko-logičke jedinke. Preko izlaznog dijela rezultati izračunavanja prenose se u okolinu.

Iz memorije *upravljačka*[3] jedinka dohvaća instrukcije i na temelju njih upravlja preostalim dijelovima računala. Upravljačka jedinka određuje koju će operaciju izvesti *aritmetičko-logička jedinka*[4].

Upravljačka jedinka i aritmetičko-logička jedinka spregnute su današnjim računalima u jednu cjelinu i, dodatno, s jednim skupom registara čine *procesor*. Procesor je automatski, strojni izvoditelj niza instrukcija koje zovemo *programom*.

Mi znamo da se današnjom mikroelektroničkom tehnologijom na milijune sićušnih tranzistora može smjestiti na jednu malu pločicu silicija, pa se pojedini spomenuti dijelovi računala danas mogu ostvariti s jednim ili tek nekoliko mikroelektroničkih sklopova — čipova. Tako danas postoje mikroelektronički *memorijski čipovi* koji sadrže desetke milijuna tranzistora. Ulazno i izlazne naprave povezuju se na računalo s pomoću mikroelektroničkih *pristupnih sklopova*. Procesori, izgrađeni s nekoliko milijuna tranzistora, smješteni su također na jednu pločicu silicija — na jedan čip veličine nekoliko kvadratnih centimetara. Procesori izvedeni na jednom mikroelektroničkom čipu dobili su naziv mikroprocesori[5], a računala izgrađena oko mikroprocesora su nazvana *mikroračunalima*. U prvo vrijeme to su bila računala skromnijih mogućnosti. Međutim, danas su sva osobna računala, pa i sva ostala računala, sastavljena od mikroelektroničkih sklopova, te se prefiks *mikro-* može i izostaviti. Mi ćemo često koristiti nazive procesor i računalo, a pritom će se podrazumijevati da se radi o mikroprocesorima i mikroračunalima.

Sabirnica računala

U funkcijskom prikazu računala na slici 1.1 vidljivo je kako su pojedini dijelovi računala međusobno povezani. Svaka od crta koja predstavlja tok podataka, instrukcija ili upravljačkih signala sastoji se od većeg broja vodiča preko kojih se moraju prenositi električki signali. Mi znamo da se dvije vrijednosti koji može poprimiti jedan bit u sklopovlju računala prikazuju kao dvije razine napona (primjerice: nula volta i pet volta). Za prijenos svakog bita potreban nam je stoga jedan vodič (jedna "žica"). Svaka crta u funkcijskom prikazu računala zbog toga predstavlja cijeli "snop vodiča". Jasno nam je da je takvo povezivanje nespretno. Stoga je osmišljen i najčešće se koristi *sabirnički sustav* povezivanja dijelova računala. Sabirnica[6] je jedan zajednički snop vodiča na koji su spojeni svi dijelovi računala. Osim vodiča sabirnica ima i svoj sabirnički elektronički sklop, koji pomaže pri ostvarivanju veza.

[3] engleski *Control Uunit*

[4] engleski *Arithmetic-Logic Unit* — ALU

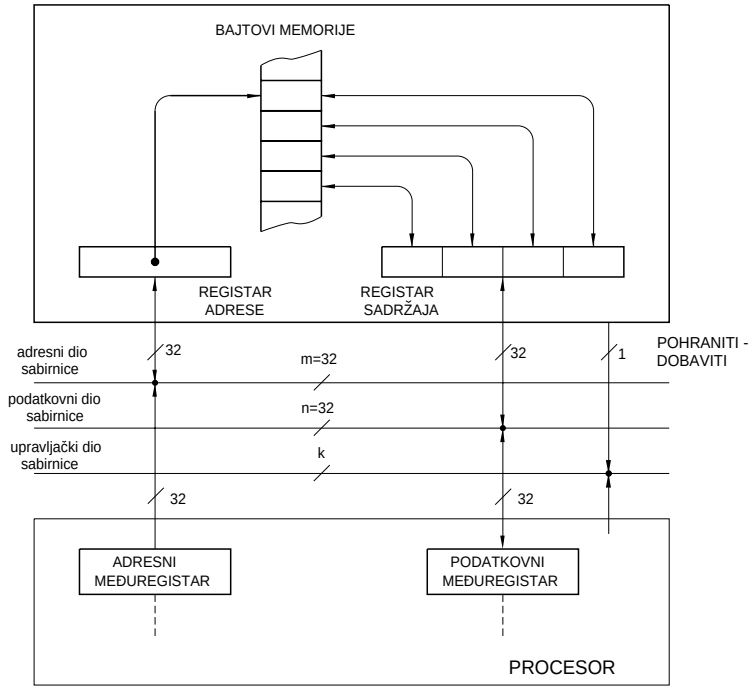
[5] prvi četverobitovni mikroprocesor proizveden je 1971. godine

[6] U engleskom jeziku je sabirnica dobila naziv *bus*, što je izvorni naziv za autobus koji sabire i razvozi putnike

- adresni dio sabirnice s m vodiča;
- podatkovni dio s n vodiča;
- upravljački dio s k vodiča.

Na slici je pretpostavljeno da je $m = 32$ i $n = 32$, tj. da koristimo tridesetdvobitovnu memorijsku adresu, te da se preko sabirnice istovremeno može prenijeti četiri bajta. Govorimo da je "širina" pristupa do memorije četiri bajta ili 32 bita.

Djelovanje i mogući sadržaj memorije razmatrat ćemo u sljedećem poglavlju. Ovdje ćemo samo ukratko reći da sklopovlje memorije ima svoj registar adrese u koji ulazi adresa sa sabirnice. U našem primjeru to će biti adresa prvog od četiri bajta koji će biti preneseni. Ovisno o tome da li se radi o pohranjivanju ili dobavljenju sadržaja iz spremnika, bajtovi se pune iz registra sadržaja ili se sadržaji bajtova prenose se u registar sadržaja memorijskog sklopovlja, što na slici 1.4 simboliziraju četiri strelice koje povezuju bajtove spremnika i registar sadržaja.



Sl. 1.4. Procesor i memorija povezani na sabirnicu

Procesor je također povezan na sabirnicu preko svojih registara. Nazvat ćemo ih međuregistrima prema sabirnici ili, kraće, samo *međuregistrima*. Iz *adresnog međuregistra* postavlja se adresa na adresni dio sabirnice, a *podatkovni međuregistar* ima "dvosmjerno" djelovanje: podaci se mogu prenositi iz procesora u memoriju ili iz memorije u procesor.

2.
Memorija računala — osnovni tipovi podataka

1. Funkcionalni opis memorije računala

Središnji je dio svakog računala *memorija*. Memorija je je ishodište i odredište svih informacija koje kolaju između svih ostalih dijelova računala. U memoriji se pohranjuju podaci, tj. memorija "pamti" — "memorira" sadržaje koji se u nju pohranjuju. Zbog toga se za memoriju sve više koristi i naziv *spremnik*. U predmetu digitalna elektronika upoznali smo mikroelektroničke sklopove za izgradnju memorijskih podsustava.

S uporabnog stanovišta memorija se može zamisliti kao skup registara jednake duljine. Pritom se oni više neće zvati registri već *riječi*[1]. U riječ se može pohraniti neki sadržaj i nakon proizvoljno dugog vremena — onda kada to bude potrebno — taj sadržaj opet pročitati. U načelu duljina riječi mogla bi biti proizvoljna, ali se pokazalo praktičnim da duljine riječi budu potencije broja dva. U današnjim mikroračunalima uobičajilo se da najmanja spremnička jedinka ima duljinu od osam bitova i dobila je izmišljeni engleski naziv — *byte*, koji mi čitamo i pišemo — *bajt*.

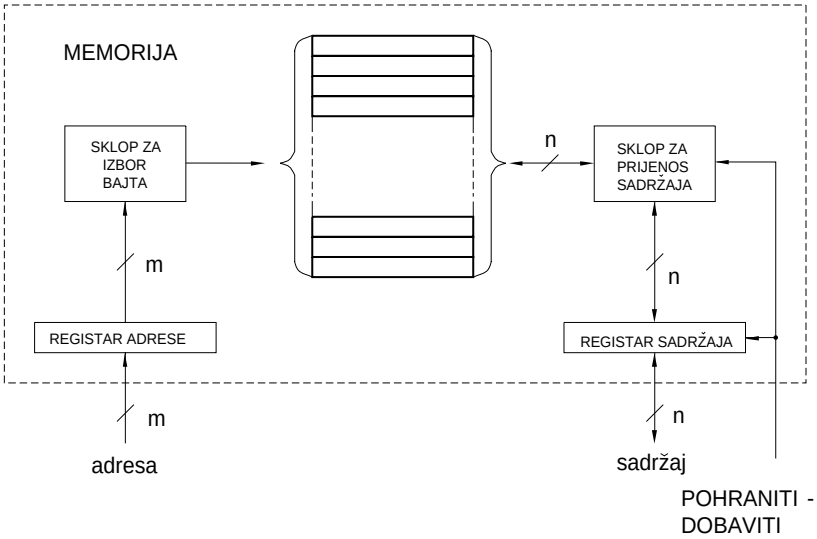
Za označavanje bajta služi kratica B (veliko slovo B). U nas se koristiti i naziv *oktet*[2] ili osmorka bitova, ali se time pretežito podrazumijeva bilo kakva nakupina od osam bitova. Isto tako, za nakupinu od četiri bita koristi se naziv *četvorka bitova* ili *kvartet*[3]. Tako se može reći da jedan bajt sadrži jedan oktet ili dva kvarteta bitova, odnosno jednu osmorku ili dvije četvorke bitova.

Do svakog bajta u memoriji može se neposredno pristupiti. Svaki bajt dobiva svoj redni broj. Taj broj je čvrsto povezan s tim bajtom, on je njegov "kućni broj" i s pomoću njega se bajt odabire. Stoga se taj broj naziva *adresom* bajta.

[1] u engleskom — *word*

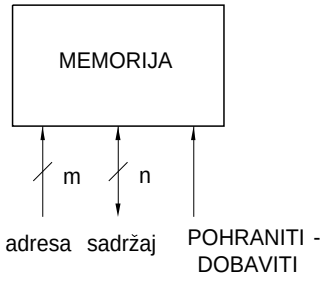
[2] u francuskom se za bajt upotrbljava naziv *octet*

[3] u engleskom se za kvartet koristi naziv *nibble* — što znači mali zalogaj



Sl. 2.2. Građa memorije

Za izvedbu memorije potrebni su još neki pomoćni elektronički sklopovi. Slika 2.2 prikazuje način izgradnje memorijskog podsustava. Registar adrese i registar sadržaja služe za uspostavljanje veze s ostalim dijelovima računala. U registar adrese mora se pri pristupu memoriji pohraniti adresa riječi. Na temelju te adrese sklop za izbor riječi odabire adresirani bajt. Ovisno o tome kako je postavljen upravljački signal na priključku označenom s POHRANITI-DOBAVITI, obavlja se preko sklopa za prijenos sadržaja odgovarajuća razmjena registra sadržaja i adresiranog bajta. Pri dobavljanju sadržaja iz memorije najprije se sadržaj adresiranog bajta smješta u registar sadržaja i iz tog se registra može prenositi dalje. Suprotno, kada se želi pohraniti neki sadržaj u memoriju, tada se on mora, istovremeno s postavljanjem adrese, staviti u registar sadržaja.



Sl. 2.3.

Memorija se može jednostavno prikazati u blokovnom obliku kao na sl. 2.3. S upotrebnog motrišta ne mora se voditi računa o tranzistorima, tranzistorskim

"memoriran". U njih se sadržaj ne može ponovno pohranjivati, već se iz njih sadržaj može "samo čitati", pa otuda dolazi i kratica *ROM*[4]. Memorija čiji se sadržaj može mijenjati, tj. u čije se bajtove može i pohranjivati sadržaje i iz nje dobavljati sadržaje označava se kraticom *RAM*[5] . Ta kratica ne potiče od opisa toga svojstva, već je uvedena onda kada su tehnološki uvjeti omogućili da se prvotne memorije, kod kojih se do pojedinih riječi moglo pristupiti samo slijedno (tj. prelazeći redom sve prethodne riječi), zamijene memorijama koje omogućuju pristup do bajtova u bilo kojem poretku. Memorije računala mogu u istom adresnom prostoru imati neka područja adresa ostvarena RAM-om, a neka ROM-om.

Mikroelektronička tehnologija omogućila je proizvodnju čipova kod kojih se na vrlo maloj površini od nekoliko kvadratnih centimetara može smjestiti na desetke milijuna bitova. Pritom su veličine pojedinih tranzistorčića dijelovi mikrometra. S takvim čipovima izgrađuju se radne memorije koje imaju od nekoliko megabajta do stotinjak megabajta. Istovremeno, takve su memorije i vrlo brze. Vrijeme između dva uzastopna pristupa do spremnika može biti svega nekoliko desetaka nanosekundi.

Memorija bi se mogla usporediti s velikim praznim papirom (ili — mnogo komada papira!) na koji možemo zapisati sve informacije koje nas zanimaju. U našim knjigama sve informacije predočene su u obliku tekstova, zapisanih *slovima abecede, dekadskim znamenkama* i *dogovorenim znakovima interpunkcije*. Nadalje, u knjigama se nalaze i slike i to u obliku crteža ili fotografija.

Što se sve pohranjuje u memoriji računala?

Po izumitelju tiskarskog stroja J. Gutenbergu[6] zadnjih pet stotina godina naziva se Gutenbergovom civilizacijskom erom. Tiskana je knjiga, naime, omogućila širenje znanja i time pomogla razvitku obrazovanja i znanosti, a time i svekolikom napretku čovječanstva. Sve znanje koje je čovječanstvo steklo tijekom svog razvitka zapisano je u knjigama.

Postavlja se pitanje možemo li to znanje zapisati u spremnike računala. Odgovor na to pitanje je pozitivan. Moramo se samo dogovoriti kako će svi znakovi biti kodirani i kako ćemo predstaviti crteže. Već kod uvođenja oktalnih i heksadekadskih znamenki naučili smo što je to kodiranje. Jednako tako, moraju se uvesti kôdovi za sve znakove koje želimo pohraniti u memoriju.

S obzirom da se u memoriju mogu zapisati samo nizovi nula i jedinica, mora se točno utvrditi što te jedinice i nule znače. Bez dodatnih informacija o tome što nizovi nula i jedinica znače ništa se ne bi moglo zaključiti o sadržaju memorije.

[4]od engleskog *Read-Only Memory* — memorija iz koje se može samo "čitati" sadržaje

[5]od engleskog *Random Access Memory* — memorija s proizvoljnim redoslijedom pristupa

[6]Nijemac Johannes Gutenberg (oko 1400.–1468.) konstruirao je tiskarski stroj i 1455. godine dovršio je tiskanje prve knjige — Biblije na latinskom jeziku s 1282 stranice u dva stupca po 42 retka

2. Zapisivanje prirodnih brojeva — prirodni binarni kôd

Način zapisivanja prirodnih brojeva u brojevnim sustavima obrađen je opširno u predmetu "Digitalna elektronika". Predstavljanje prirodnog broja u spremniku računala slijedi sva pravila zapisivanja koja smo već izučili. Jedina posebnost koja se pri pohranjivanju u spremnik mora uvažiti je *ograničeni broj bitova* koji se koriste za zapisivanje. S obzirom da je taj način zapisivanja najprirodniji način gledanja na niz bitova, ovaj način zapisivanja prirodnog broja dobio je naziv *prirodni binarni kôd*[7].

Tako se, primjerice, u jedan bajt može zapisati brojeve s najviše osam bitova, tj. brojeve od

$$0 = 00000000\text{B}$$

do

$$255 = 11111111\ \text{B}.$$

Na papiru se u načelu ne zapisuju nule s lijeve strane. Međutim, ako se na papiru želi zapisati *sadržaj jednog bajta*, onda je uobičajeno, a i praktično *zapisati vrijednosti svih osam bitova*.

Primjer 2.2. Prikaz nekih prirodnih brojeva u jednom bajtu glasi:

$$\begin{aligned} 1 &= 00000001\ \text{B} \\ 2 &= 00000010\ \text{B} \\ 4 &= 00000100\ \text{B} \\ 8 &= 00001000\ \text{B} \\ 16 &= 00010000\ \text{B} \\ 32 &= 00100000\ \text{B} \\ 64 &= 01000000\ \text{B} \\ 128 &= 10000000\ \text{B} \end{aligned}$$

Treba zapaziti da su odabrani brojevi potencije broja 2. Ti brojevi imaju samo jednu jedinicu na odgovarajućem težinskom mjestu. □

[7] u engleskom jeziku: *Natural Binary Code* — NBC

U jedan bajt mogu se pohraniti brojevi u rasponu od 0 do 255, što nije dovoljno za mnoge zadatke koje želimo obavljati. Veći prirodni brojevi mogu se zapisati u dva ili više uzastopnih bajtova. U današnjim se računalima, ovisno o tipu računala, koriste registri duljine 8, 16, 32 ili 64 bita, pa prema tome i brojevi koji imaju toliki broj bitova. Stoga i u memoriji treba predvidjeti načine smještanja sadržaja takvih duljina.

Prirodni brojevi duljine 16 bitova mogu se smjestiti u dva uzastopna bajta, oni duljine 32 bita u četiri uzastopna bajta, a brojevi duljine 64 bita u nakupinu uzastopnih osam bajtova. Pokazalo se praktičnim da se za *adresiranje čitave nakupine bajtova* koristi *adresa bajta s najmanjom adresom* i, isto tako, da je prikladno te nakupine bajtova smjestiti u spremnik "poravnato" tj. tako da se:

- prvi bajt skupine od dva bajta smjesti na parnu adresu (cjelobrojni višekratnik od dva) — ta adresa završava s 0;
- prvi bajt skupine od četiri bajta smjesti na adresu koja je cjelobrojni višekratnik od četiri — ta adresa završava s 00;
- prvi polubajt skupine od osam bajtova smjesti na adresu koja je cjelobrojni višekratnik od osam — ta adresa završava s 000.

Sve što je ovdje rečeno o smještaju prirodnih brojeva vrijedi i za smještanje drugih tipova podataka u spremniku.

U svezi sa smještanjem podataka u nakupinu bajtova mora se uvažiti još jedan dogovor. Naime, sadržaj zapisan u spremniku mora se pri njegovu korištenju prenijeti u registre izvan spremnika. Kada se prenose nakupine bajtova, mora se utvrditi način popunjavanja registara.

Razmotrimo to na primjeru prijenosa dva uzastopna bajta iz spremnika u 16-bitovni registar. S jednim bajtom popunjava se jedna polovina registra, a s drugim bajtom druga polovina. Postavlja se pitanje kojim redoslijedom se bajtovi smještaju u registar. Moguća su dva načina smještanja:

- bajt s manjom (parnom) adresom smješta se u osam bitova manje vrijednosti ("desnije bitove", "donju polovinu") registra, a sljedeći bajt (tj. bajt čija je adresa za jedan veća) smješta se u osam bitova veće vrijednosti ("lijevije bitove"; "gornju polovinu") registra;
- bajt s manjom (parnom) adresom smješta se u osam bitova veće vrijednosti ("lijevije bitove", "gornju polovinu") registra, a sljedeći bajt (tj. bajt čija adresa je za jedan veća) smješta se u osam bitova manje vrijednosti ("desnije bitove"; "donju polovinu") registra.

U različitim računalima upotrebljavaju se ili jedan ili drugi način pohranjivanja, a kod nekih se čak može birati način. Pri uobičajenom korištenju računala način smještanja uopće nije bitan. On dolazi do izražaja u nekim posebnim primjenama. Važno je, međutim, da se to punjenje, a i pohranjivanje u bajtove memorije uvijek obavlja istim redoslijedom[8].

[8]U engleskom jeziku se za način smještanja kod kojeg se na manjoj adresi nalaze "manje vrijedni" bitovi

U poluriječ se, dakle, može zapisati najveći broj

$$2^{16} - 1 = 65535,$$

u riječ se može pohraniti najveći broj

$$2^{32} - 1 = 4\,294\,967\,295,$$

a u dvostruku riječ najveći broj

$$2^{64} - 1 = 18\,446\,744\,073\,709\,551\,615.$$

Prva dva broja mogu se izračunati s pomoću džepnih kalkulatora koji prikazuju deset znamenaka. Međutim, zadnji broj ima dvadeset mjesta i može se kalkulatorom samo približno izračunati pa će biti prikazan u eksponencijalnom obliku.

Približno se brojevi 2^{32} i 2^{64} (kod približnog zapisivanja nema nikakvog smisla govoriti o brojevima $2^{32} - 1$ i $2^{64} - 1$, jer oduzimanje jedinice kod tako velikih brojeva nema nikakvog značenja) mogu zapisati kao

$$2^{32} \cong 4\,000\,000\,000 = 4 \cdot 10^9 \quad \text{i}$$
$$2^{64} \cong 18\,000\,000\,000\,000\,000\,000 = 18 \cdot 10^{18}.$$

Do približnih veličina ovih brojeva možemo doći i na drugi način. Već smo naučili da je $2^{10} = 1024 \cong 1000 = 10^3$.

Stoga se može pisati

$$2^{32} = 2^2 \cdot 2^{30} = 2^2 \cdot (2^{10})^3 \cong 4 \cdot (10^3)^3 = 4 \cdot 10^9$$

i

$$2^{64} = 2^4 \cdot 2^{60} = 2^4 \cdot (2^{10})^6 \cong 16 \cdot (10^3)^6 = 16 \cdot 10^{18} = 16 \cdot 10^9 \cdot 10^9.$$

Prema tome, broj 2^{32} je približno jednak četiri milijarde. Za broj 2^{64} na ovaj se način dobiva približna vrijednost od šesnaest milijarda milijardi, a on je manji od 18 milijarda milijardi, kao što se vidi iz precizno izračunatog ranije navedenog broja. Zbog jednostavnosti ćemo koristiti približnu vrijednost $16 \cdot 10^{18}$.

Primjer 2.6. Počevši od adrese 2000 H u spremniku su u osam uzastopnih bajtova pohranjeni sadržaji prikazani na slici 2.6. Napišite te sadržaje, najprije u heksadekadskom obliku a zatim u dekadskom obliku, ako oni predstavljaju:

a) četiri 16–bitovna prirodna broja s bitovima manje vrijednosti na manjim adresama;
b) dva 32–bitovna prirodna broja s bitovima manje vrijednosti na manjim adresama;
c) četiri 16–bitovna prirodna broja s bitovima veće vrijednosti na manjim adresama;
d) dva 32–bitovna prirodna broja s bitovima veće vrijednosti na manjim adresama.

Ponovimo pravila računanja u binarnom sustavu koja smo naučili izučavajući digitalnu elektroniku. Operacije zbrajanja i oduzimanja možemo razjasniti tablicama koje vrijede za jednoznamenkaste brojeve.

Tablica zbrajanja 2.1. a) prikazuje pravilo zbrajanja dvaju bitova x_i i y_i. Bit zbroja neka se zove s_i. Zbroj $1+1=10$ (jedan više jedan je dva) ne može se zapisati kao jedan bit već su za to potrebna dva bita. To je važno uočiti i primijeniti kod zbrajanja višebitovnih brojeva. Znamenka drugog bita zbroja mora se pribrojiti sljedećem lijevom bitu (podsjetite se kako se to čini u dekadskom zapisu!). To, međutim, znači da će pri zbrajanju trebati pribrajati tri jednobitovna broja. U tom zbroju mogu se dobiti zbrojevi od nula do najviše tri (kada sva tri pribrojnika imaju vrijednost jedan).

Kada zbroj iznosi dva ili tri (to se zapisuje kao 10 i 11), vrijednost 1 se prenosi na prvo lijevo više binarno mjesto. Taj bit se naziva bit prijenosa i može se označiti s c_i. Tablica 2.1. b) prikazuje zbroj triju jednobitovnih brojeva x_i, y_i i c_i. Rezultat se može zapisati kao bit zbroja s_i i bit prijenosa na c_{i+1}. Indeks $i+1$ označava da se taj bit prenosi na sljedeće više binarno mjesto.

Tablica 2.1.

a) zbroj dvaju jednobitovnih brojeva

x_i	y_i	s_i
0	0	0
0	1	1
1	0	1
1	1	10

b) zbroj triju jednobitovnih brojeva

c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Pogled na tablicu 2.1. b) ukazuje da je bit zbroja s_i jednak jedinici kada u tri pribrojnika postoji neparan broj jedinica i jednak nuli kada se u tri pribrojnika nalazi neparan broj jedinica. Prijenos na sljedeći bit c_{i+1} jednak je nuli kada u pribrojnicima postoje manje od dvije jedinice, a jednak je jedinici kada se u pribrojnicima nalaze dvije ili tri jedinice.

Primjer 2.7. Izračunati zbroj $X+Y=10111011+101110$. Brojeve treba napisati jedan ispod drugog tako da bitovi jednake težine budu poravnati. Pribrojnik s manje bitovnih mjesta može se nadopuniti nulama s lijeve strane, što mu ne mijenja vrijednost. Zbrajanje se započinje od krajnjeg desnog bita. Taj se bit može zvati nultim bitom jer ima indeks jednak nuli. Prijenos u nulti bit ne postoji (jer desno od njega više nema bitova), pa se može reći da je on jednak nuli, tj. $c_0=0$. Bit zbroja se potpisuje ispod vodoravne crte. Bitovi prijenosa u sljedeće

Brojeve treba napisati jedan ispod drugoga tako da bitovi jednake težine budu poravnati. Umanjenik se može nadopuniti nulama lijevo od njegovog najvišeg bita. Oduzimanje se započinje s nultim bitovima s tim da je $p_0 = 0$. Bitovi razlike zapisuju se ispod vodoravne crte, a bitovi posudbe koji se prenose s višeg binarnog mjesta zapisani su iznad bitova umanjenika i označeni znakom $\rightarrow$.

bitovi posudbe		$0\rightarrow$	$0\rightarrow$	$0\rightarrow$	$1\rightarrow$	$1\rightarrow$	$0\rightarrow$	$0\rightarrow$	0
X		1	0	1	1	1	0	1	1
Y	$-$	0	0	1	0	1	1	1	0
razlika		1	0	0	0	1	1	0	1

Ispravnost oduzimanja može se provjeriti tako da se vrijednosti umanjenika, umanjitelja i razlike zapišu u dekadskom sustavu:

$$\begin{array}{rr} & 10111011_2 = \quad 187_{10} \\ - & 101110_2 = \quad 46_{10} \\ \hline & 10001101_2 = \quad 141_{10} \end{array}$$

□

Primjer 2.9. Izračunajte razliku $101110 - 10111011$.

Treba uočiti da je umanjenik manji od umanjitelja i razlika neće pripadati skupu prirodnih brojeva. Oduzimanje se provodi prema pravilima objašnjenim u primjeru 2.8:

bitovi posudbe		$1\rightarrow$	$1\rightarrow$	$1\rightarrow$	$1\rightarrow$	$1\rightarrow$	$1\rightarrow$	$1\rightarrow$	0
X		0	0	1	0	1	1	1	0
Y	$-$	1	0	1	1	1	0	1	1
razlika		0	1	1	1	0	0	1	1

Ovdje umanjenik nadopunjujemo nulama lijevo od najvišeg bita različitog od nule tako da se za sve bitove mogu koristiti pravila tablice 2.2. b). Međutim, u postupku oduzimanja pojavljuje se posudba iz nepostojećeg bita, što znači da oduzimanje nije dozvoljeno i dobivena razlika se ne smije uvažiti. □

Pri odabiru broja bitova za prikaz brojeva treba biti oprezan. Kada brojeve zapisujemo na papiru, brojevna mjesta možemo proizvoljno povećavati tijekom izračunavanja. Međutim, kada odaberemo jedan od načina zapisivanja broja u memoriji *broj binarnih mjesta je ograničen* i ne može se više povećavati.

Stoga se pri obavljanju aritmetičkih operacija zbrajanja i oduzimanja mora dodatno paziti da se dobiveni rezultat može zapisati u odabrani broj bitova. Pri zbrajanju dvaju brojeva mogao bi nastati prijenos u krajnje lijevi nepostojeći bit zapisa, a pri oduzimanju posudba iz krajnje lijevog nepostojećeg bita. Mogli bismo zamisliti da taj krajnje lijevi bit (ovisno o duljini zapisa: deveti, sedamnaesti,

$C = 1$, dakle, označava da je rezultat neispravan. Temeljem pravila oduzimanja dobiva se:

bitovi posudbe	**0**→	0→	0→	0→	0→	0→	0→	1→	0	
7EH		0	1	1	1	1	1	1	0	B
51H	–	0	1	0	1	0	0	0	1	B
razlika		0	0	1	0	1	1	0	1	B

C = 0, rezultat je ispravan;

bitovi posudbe	**1**→	1→	1→	1→	1→	1→	1→	0→	0	
51H		0	1	0	1	0	0	0	1	B
7EH	–	0	1	1	1	1	1	1	0	B
razlika		1	1	0	1	0	0	1	1	B

C = 1, rezultat je neispravan;

bitovi posudbe	**1**→	1→	0→	1→	1→	1→	0→	0→	0	
A3H		1	0	1	0	0	0	1	1	B
C5H	–	1	1	0	0	0	1	0	1	B
razlika		1	1	0	1	1	1	1	0	B

C = 1, rezultat je neispravan;

bitovi posudbe	**0**→	0→	0→	0→	0→	0→	0→	0→	0	
B0H		1	0	1	1	0	0	0	0	B
A0H	–	1	0	1	0	0	0	0	0	B
razlika		0	0	0	1	0	0	0	0	B

C = 0, rezultat je ispravan.

□

3. Pohranjivanje cijelih brojeva

2.3.1. Kôdovi za zapisivanje cijelih brojeva

Skup cijelih brojeva $\mathbf{Z}$ čine negativni cijeli brojevi $-1, -2, -3, \ldots$, broj 0, te pozitivni cijeli brojevi $1, 2, 3, \ldots$. Skup pozitivnih cijelih brojeva jednak je skupu prirodnih brojeva $\mathbf{N}$. Za skup $\mathbf{N}_0$, koji uključuje prirodne brojeve i broj 0, može se reći da je jednak skupu nenegativnih cijelih brojeva.

Kada se cijeli brojevi zapisuju na papiru, tada se ispred negativnog broja stavlja predznak minus, dok se ispred broja 0 i pozitivnih brojeva može pisati, ali se obično izostavlja, predznak plus. Može se ustanoviti da se cijeli brojevi zapisuju tako da im se zapiše apsolutna vrijednost i ispred nje predznak " – " za negativne

pohranjivanja postoje dvije nule: -0 i $+0$, tj. nula se može zapisati na dva načina.

U takvom se zapisu brojevi lako prepoznaju. Pritom, naravno, treba znati da se radi o *cijelim* a ne o prirodnim brojevima. Pretvorba u dekadski zapis obavlja se na isti način kao i kod prirodnih brojeva, ali se koristi samo $n-1$ bitova. Ispred pretvorenog zapisa se tada stavlja znak $+$ (ili se taj predznak uopće ne piše) ako se u bitu predznaka nalazi 0, odnosno piše se predznak $-$ ako se u bitu predznaka nalazi 1.

Međutim, operacije zbrajanja i oduzimanja su malo složenije, jer treba voditi računa o predznacima operanada. Tako se npr. operacija zbrajanja obavlja sljedećim algoritmom:

```
usporediti predznake pribrojnika;
ako je ustanovljena jednakost pribrojnika
    onda
        zbrojiti apsolutne vrijednosti pribrojnika;
        ispred zbroja staviti predznak pribrojnika
    inače
        od veće apsolutne vrijednosti oduzeti manju
        apsolutnu vrijednost;
        ispred razlike staviti predznak pribrojnika s
        većom apsolutnom vrijednošću;
```

Možemo se lako uvjeriti da se oduzimanje obavlja sličnim, još malo složenijim algoritmom.

Pokazuje se da se operacije zbrajanja i oduzimanja mogu znatno pojednostavniti ako se za kodiranje cijelih brojeva upotrijebi malo složeniji kôd. U tom kôdu je bitu predznaka *pripisana vrijednost* -2^{n-1}, tj. najviši bit ima *negativni težinski faktor*. Prema tome, niz bitova $b_{n-1}b_{n-2}b_{n-3}\ldots b_2b_1b_0$ jedne riječi predstavlja cijeli broj

$$B = -b_{n-1}\cdot 2^{n-1} + b_{n-2}\cdot 2^{n-2} + b_{n-3}\cdot 2^{n-3} + \ldots + b_2\cdot 2^2 + b_1\cdot 2^1 + b_0\cdot 2^0.$$

Ako je najviši bit jednak nuli, tj. $b_{n-1} = 0$, što je dogovorena oznaka pozitivnog cijelog broja, onda preostali niži bitovi mogu biti pročitani kao prirodni broj s $n-1$ bitova. Za taj broj može se uvesti oznaka B', te je:

$$B' = b_{n-2}\cdot 2^{n-2} + b_{n-3}\cdot 2^{n-3} + \ldots + b_2\cdot 2^2 + b_1\cdot 2^1 + b_0\cdot 2^0.$$

Prema tome pozitivni cijeli brojevi se zapisuju na potpuno jednaki način kao i u prethodnom načinu zapisivanja s predznakom i apsolutnom vrijednošću. Međutim, kada je $b_{n-1} = 1$, tada vrijednosti B' treba pribrojiti vrijednost težinskog faktora -2^{n-1} (tj. oduzeti vrijednost 2^{n-1}), pa je:

$$B = B' - 2^{n-1}.$$

Zbroj dvaju suprotnih brojeva, tj. brojeva različita predznaka s istim apsolutnim vrijednostima, jednak je nuli. Tako je, npr.

$$2 + (-2) = 0, \quad 126 + (-126) = 0.$$

Pogledajmo što se dobiva ako se po pravilima zbrajanja za prirodne binarne brojeve zbroje dva suprotna broja (s tim da je negativni broj zapisan u obliku koji ima negativni težinski faktor u bitu predznaka), a *pravila zbrajanja primjenjuju se i na bit predznaka*. Pretpostavimo da se brojevi zapisuju u jedan bajt, tj. da je $n = 8$. Zbrajanje daje:

$$\begin{array}{r} 2 \\ +(-2) \\ \hline 0 \end{array} \qquad \begin{array}{r} 00000010\,\mathrm{B} \\ +\,11111110\,\mathrm{B} \\ \hline 1\,|\,00000000\,\mathrm{B} \end{array}$$

$$\begin{array}{r} 126 \\ +(-126) \\ \hline 0 \end{array} \qquad \begin{array}{r} 01111110\,\mathrm{B} \\ +\,10000010\,\mathrm{B} \\ \hline 1\,|\,00000000\,\mathrm{B} \end{array}$$

Rezultat zbrajanja uvijek izaziva prijenos u dodatni bit lijevo od bita predznaka. Ako se taj broj pročita kao prirodni broj on je jednak

$$100000000\,\mathrm{B} = 2^8 = 256.$$

Općenito, ako se s B_1 označi pozitivni broj s n bitova i s B_2 njemu suprotni negativni broj, uvijek vrijedi

$$B_1 + B_2 = 2^n$$

ili

$$B_2 = 2^n - B_1.$$

S obzirom da je broj B_1 jednak apsolutnoj vrijednosti broja B_2, to je $B_2 = 2^n - \mathrm{abs}(B_2)$.

Odatle slijedi algoritam za dobivanje zapisa negativnog broja:

> `oduzeti apsolutnu vrijednost zadanog broja od` 2^n;
> `dobiveni rezultat prevesti u binarni zapis.`

Primjer 2.13. Brojeve -73 i -39 treba zapisati u osmerobitovnom kôdu s negativnom vrijednošću težinskog faktora predznaka. Slijedom opisanog algoritma dobiva se:

$$\begin{aligned} 2^8 - 73 &= 256 - 73 = 183, \\ 183 &= 10110111\,\mathrm{B}, \\ -73 &= 10110111\,\mathrm{B} \end{aligned}$$

i

odnosno:

$$\begin{aligned} 39 &= && 00100111\,\text{B} \\ 2^8-1 &= && 11111111\,\text{B} \\ & && -00100111\,\text{B} \\ \hline & && 11011000\,\text{B} \\ & && +\qquad 1\,\text{B} \\ \hline -39 &= && 11011001\,\text{B} \end{aligned}$$

Treba zapaziti da se oduzimanje od 11111111 dobije tako da se u binarnom zapisu umanjenika naprosto nule zamijene jedinicama i obrnuto, jedinice nulama. □

Komplement i dvojni komplement

U svezi s gore opisanom pretvorbom mogu se uvesti još neki nazivi koji se u računarstvu koriste. Naime, ako se prethodna razmatranja svedu na jednobitovne brojeve, tj. u izraz $B_1 + B_2 = 2^n$ uvrsti $n = 1$, dobiva se $B_1 + B_2 = 2$, ili $B_2 = 2 - B_1$.

Može se reći da je B_2 dopuna od B_1 do dva ili *komplement*[11] prema dva. Kraće se kaže da je B_2 *dvojni komplement* od B_1.

Oduzimanjem i pribrajanjem jedinice na desnoj strani gornji se izraz može preurediti u $B_2 = ((2-1) - B_1) + 1$, ili $B_2 = (1 - B_1) + 1$.

Broj $1 - B_1$ može se označiti s B_2' i nazvati taj broj *komplementom prema jedan* ili kraće samo *komplementom* broja B_1. Prema tome, dobiva se $B_2' = 1 - B_1$, odnosno $B_2 = B_2' + 1$.

Uz $n = 1$ su B_1 i B_2' binarne znamenke i mogu poprimiti samo vrijednosti 0 ili 1. Za binarne znamenke, dakle, vrijedi:

komplement od 0 je 1,
komplement od 1 je 0.

Uvedeni nazivi mogu se proširiti i na n-terobitovne brojeve. Tako je komplement B_2' broja B_1 jednak $B_2' = (2^n - 1) - B_1$, a dvojni komplement B_2 broja je $B_2 = 2^n - B_1$.

Između ta dva komplementa postoji veza $B_2 = B_2' + 1$.

S ovim novim nazivima može se drugim riječima prepričati već opisani algoritam za prevođenje negativnog cijelog broja u dvokomplementni binarni zapis. On glasi:

```
zapisati u prirodnom binarnom kôdu
apsolutnu vrijednost broja;
napisati komplement tog zapisa;
komplementu dodati jedan;
```

[11] komplement dolazi od latinskog *complementum* — dopuna, upotpunjavanje

i osnovne operacije. Pritom se, prema potrebi, u nekim programskim jezicima može odabrati da li će se za pohranjivanje cijelog broja koristiti poluriječ, riječ ili dvostruka riječ, slično kao i kod pohranjivanja prirodnih brojeva. Tada se mogu koristi nazivi *kratki cijeli broj*, *cijeli broj* (pritom se misli na uobičajeno dugi cijeli broj, a to se posebno ne naglašava) ili *dugi cijeli broj*[13].

Na kraju, još jednom treba upozoriti da je jedan bit potrošen za zapisivanje predznaka, te da se preostali bitovi, njih $n-1$ u n-terobitovnom zapisu, koriste za zapisivanje vrijednosti. S n bitova može se zapisati 2^n različitih sadržaja. Polovina od njih, tj. njih 2^n-1 ima u najvišem bitu zapisanu nulu i to su pozitivni brojevi. Broj nula se zapisuje tako da su mu svi bitovi jednaki nuli. Time je nula uvrštena među pozitivne brojeve, te je najveći pozitivni broj koji se može zapisati onaj sa svim jedinicama (osim bita predznaka), tj. broj $2^{n-1}-1$. Druga polovina zapisa, koji započinju s bitom predznaka jednakim jedan, predstavlja negativne brojeve. S obzirom da među njima nema nule, svi su oni iskorišteni za negativne brojeve kojih se može zapisati ukupno 2^{n-1} (to su brojevi od -2^{n-1} do -1). Tako je, u dvokomplementnom zapisu, apsolutna vrijednost najmanjeg negativnog broja za jedan veća od najvećeg pozitivnog broja.

Primjer 2.16. Koji se cijeli brojevi mogu zapisati u poluriječ, riječ i dvostruku riječ, odnosno unutar kojeg intervala se nalaze kratki cijeli brojevi, cijeli brojevi i dugi cijeli brojevi?

Kratki cijeli brojevi pohranjuju se u dva bajta, cijeli brojevi u četiri bajta a dugi cijeli brojevi u osam bajtova. Prema tome, oni imaju 16, 32 i 64 bitova. Raspon cijelih brojeva koji se mogu tim bitovima zapisati prikazan je tablično:

broj bajtova	broj bitova	najmanji cijeli broj	najveći cijeli broj
	n	-2^{n-1}	$2^{n-1}-1$
2	16	-32768	32767
4	32	-2147483648	2147483647
8	64	-9223372036854775808	9223372036854775807

Za $n=64$ apsolutna vrijednost najmanjeg cijelog broja koji se može zapisati je 2^{63}, što je polovina od 2^{64}. Najveći pozitivni cijeli broj koji se može zapisati je $2^{63}-1$. □

Kôd s posmakom

Uz spomenute načine zapisivanja cijelih brojeva (apsolutna vrijednost s predznakom, komplement, dvojni komplement), koriste se za zapisivanje cijelih brojeva još i neki drugi kôdovi. Jedan od njih je kôd s posmakom od 2^{n-1} ili kraće: *kôd*

[13] u engleskom: *short integer, integer, long integer*

brojčanu vrijednost, s njim se može takoder mogu obavljati aritmetičke operacije. Podsjetimo se da je interpretacija binarnog zapisa $b_{n-1}b_{n-2}b_{n-3}\ldots b_1b_0$ jednaka

$$N = b_{n-1}\cdot 2^{n-1} + b_{n-2}\cdot 2^{n-2} + b_{n-3}\cdot 2^{n-3} + \ldots + b_1\cdot 2^1 + b_0\cdot 2^0.$$

Zbog jednostavnosti se za objašnjavanje pravila zbrajanja i oduzimanja može upotrijebiti četverobitovna riječ, tj. kvartet bitova. Stvoreni zaključci se zatim mogu poopćiti za riječi proizvoljne duljine. U tablici 2.4 predstavljeni su svi brojevi koji se mogu zapisati u jedan kvartet kao i njihovi dekadski ekvivalenti i to redom od najmanjeg do najvećeg broja.

Tablica 2.4. *Četverobitovni cijeli brojevi*

dvokomplementni zapis	dekadski zapis
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Ako se započne s najmanjim brojem $1000\ \mathrm{B} = -8$, tada se postepenim pribrajanjem broja 1 dolazi nakon sedam pribrajanja do broja $1111\ \mathrm{B} = -1$ (tim se pribrajanjem u tablici 2.4 postepeno silazi redak po redak). Pribrajanjem broja 1 broju $1111\ \mathrm{B}$ po pravilima zbrajanja za prirodne binarne brojeve dobiva se

$$\begin{array}{rr} -1 & 1111\ \mathrm{B} \\ +1 & +\quad 1\ \mathrm{B} \\ \hline 0 & \mathbf{1}|\ 0000\ \mathrm{B}, \end{array} \qquad \text{uz } \mathrm{C} = 1.$$

Kada se zanemari bit prijenosa C, tada i to pribrajanje daje ispravan rezultat. Sada se može nastaviti uzastopno pribrajati jedinica dok se ne dođe do broja

ispred apsolutnih vrijednosti odgovarajuće predznake. Sve mogućnosti koje se mogu pojaviti pri zbrajanju ili oduzimanju tih cijelih brojeva svode se na rezultat čija će apsolutna vrijednost biti razlika ili zbroj apsolutnih vrijednosti X i Y. Postoji ukupno osam mogućnosti:

$$\begin{aligned}
(+X) + (+Y) &= +(X+Y),\\
(-X) + (+Y) &= -(X-Y),\\
(+X) - (+Y) &= +(X-Y),\\
(-X) - (+Y) &= -(X+Y),\\
(+X) + (-Y) &= +(X-Y),\\
(-X) + (-Y) &= -(X+Y),\\
(+X) - (-Y) &= +(X+Y),\\
(-X) - (-Y) &= -(X-Y).
\end{aligned}$$

S obzirom da se pretpostavlja da se X i Y mogu zapisati u zadani broj bitova, to će se sigurno moći zapisati i brojevi manje apsolutne vrijednosti $X-Y$. Međutim, može se dogoditi da se brojevi $(X+Y)$ i $-(X+Y)$, više ne mogu zapisati u taj broj bitova.

Prema tome, može se zaključiti da preliva neće biti kada se zbrajaju brojevi različitih predznaka ili odbijaju brojevi jednakih predznaka. Opasnost preliva postoji kada se zbrajaju brojevi jednakih predznaka i oduzimaju brojevi različitih predznaka.

Zbog jednostavnosti razmotrimo samo operaciju zbrajanja. Pokazuje se da je rezultat ispravan kada:

- pribrojnici imaju različite predznake ili
- pribrojnici imaju jednake predznake i predznak dobivenog zbroja je jednak predznacima pribrojnika.

Rezultat je neispravan kada:

- pribrojnici imaju jednake predznake a u zbroju se pojavljuje suprotni predznak.

Označimo n-bitovne pribrojnike s A i B, a njihov zbroj, koji je dobiven zbrajanjem po pravilima za prirodni binarni kôd sa S, tj.

$$\begin{aligned}
A &= a_{n-1}a_{n-2}a_{n-3}\ldots a_1a_0,\\
B &= b_{n-1}b_{n-2}b_{n-3}\ldots b_1b_0,\\
S &= s_{n-1}s_{n-2}s_{n-3}\ldots s_1s_0.
\end{aligned}$$

Predznaci pribrojnika su a_{n-1} i b_{n-1}, a predznak zbroja s_{n-1}. Algoritam za određivanje vrijednosti OV mogao bi izgledati ovako:

zapisanih u konačni broj bitova, u računalima se upotrebljavaju i neke druge zastavice.

Tako se npr. upotrebljava još jedan dodatna zastavica s oznakom ZF[15], koja označava da li je rezultat neke operacije jednak nuli. Zastavica se postavlja u $\mathrm{ZF} = 1$ kada je rezultat operacije jednak nuli, a u $\mathrm{ZF} = 0$ kada je rezultat različit od nule.

Daljnja korisna zastavica je SF[16], koja se postavlja u skladu s predznakom rezultata operacije. $\mathrm{SF} = 0$ kada je rezultat operacije pozitivan, a $\mathrm{SF} = 1$ kada je rezultat negativan. Drugim riječima, vrijednost zastavice SF jednaka je vrijednosti bita predznaka.

Nadalje, nekada je korisno znati da li u rezultatu neke operacije postoji parni ili neparni broj jedinica. Zato služi tzv. zastavica parnosti PF ili samo P[17]. Kada je broj jedinica u rezultatu paran, postavlja se $\mathrm{P} = 0$, a kada je broj jedinica neparan, postavlja se $\mathrm{P} = 1$.

Za neke primjene potrebno je znati da li je nastao prijenos iz donje u gornju polovinu bajta. To se označava zastavicom HF ili AC[18]. Zastavica se postavlja u $\mathrm{AC} = 1$ kada nastane prijenos iz donje polovine bajta, a inače je $\mathrm{AC} = 0$.

Svih šest zastavica C, OV, ZF, SF, P i AC postavljaju se u odgovarajuće vrijednosti kod svake aritmetičke operacije, a promatraju se samo one koje želimo promatrati. Tako se npr. pri obavljanju operacija s cijelim brojevima promatra samo zastavica OV, a stanje zastavice C se ne promatra. U raznim računalima zastavice se mogu drugačije zvati, a mogu postojati i još neke druge dodatne zastavice.

Primjer 2.18. Napišite vrijednosti šest zastavica nakon obavljanja sljedećih aritmetičkih operacija: $3\mathrm{F\,H} + 1\mathrm{A\,H}, 4\mathrm{E\,H} + \mathrm{A6\,H}, 28\mathrm{\,H} + \mathrm{D8\,H}$ i $72\mathrm{\,H} + 8\mathrm{C\,H}$. Pretpostavite da se koristi osmerobitovni prirodni binarnoi kôd, odnosno dvokomplementni osmerobitovni zapis cijelih brojeva

Dobiva se redom:

$$\begin{array}{rr} 3\mathrm{F\,H} & 00111111\,\mathrm{B} \\ +1\mathrm{A\,H} & +\ 00011010\,\mathrm{B} \\ \hline 59\,\mathrm{H} & 01011001\,\mathrm{B} \end{array}$$

te je $\mathrm{C} = 0, \mathrm{OV} = 0, \mathrm{ZF} = 0, \mathrm{SF} = 0, \mathrm{P} = 0, \mathrm{AC} = 1$;

$$\begin{array}{rr} 4\mathrm{E\,H} & 01001110\,\mathrm{B} \\ +\mathrm{A6\,H} & +\ 10100110\,\mathrm{B} \\ \hline \mathrm{F4\,H} & 11110100\,\mathrm{B} \end{array}$$

[15] kratica ZF dolazi od engleskog *zero flag* — zastavica nule

[16] kratica SF dolazi od engleskog *sign flag* — zastavica predznaka

[17] kratica PF dolazi od engleskog *parity flag* — zastavica parnosti

[18] kratica HF dolazi od engleskog *half flag* — zastavica polovine, a AC od engleskog *auxiliary carry flag*

jedan kvartet bitova (tri bita nisu dovoljna jer se s pomoću njih može zapisati samo osam brojeva). Pritom nam od mogućih šesnaest vrijednosti kvarteta treba samo deset, kao što je prikazuje tablica 2.5.

Tablica 2.5. *Kodiranje BCD znamenki*

sadržaji kvarteta	BCD znamenke
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	ne koristi se
1011	ne koristi se
1100	ne koristi se
1101	ne koristi se
1110	ne koristi se
1111	ne koristi se

Prvih deset heksadekadskih znamenki jednake su dekadskim znamenkama 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9, a za obilježavanje preostalih šest znamenki koristimo slova A, B, C, D, E i F.

Pakiranje BCD brojeva u bajtove

Pohranjivanjem samo jedne znamenke u jedan bajt memorije bila bi iskorištena samo jedna polovina raspoloživih bitova. Stoga je uobičajeno “pakirati” po dvije znamenke u jedan bajt memorije. Drugim riječima, u jedan bajt tako se može pohraniti dvoznamenkasti dekadski broj. Pritom se pretpostavlja da je znamenka jedinica pohranjena u donji kvartet i znamenka desetica u gornji kvartet bajta. Za pohranjivanje višeznamenkastih brojeva treba uzeti više uzastopnih bajtova.